

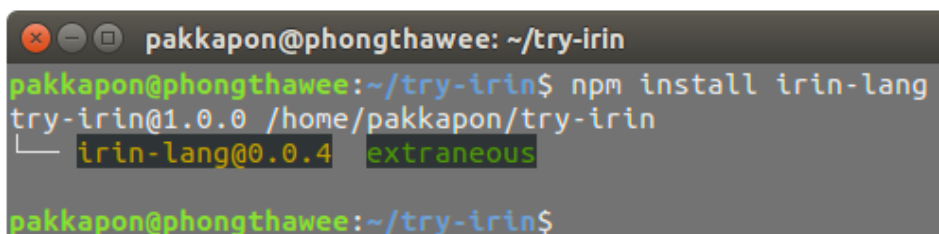
คู่มือการติดตั้งอย่างละเอียด

การดาวน์โหลด

ท่านสามารถติดตั้งตัวแปลภาษาไอรินเพื่อใช้กับโปรแกรมของท่านได้โดยการติดตั้งได้ผ่านทาง npm, bower หรือสามารถติดตั้งผ่านการดาวน์โหลดไฟล์ที่ถูกแปลงเป็นจาวาสคริปและลดขนาดแล้วได้

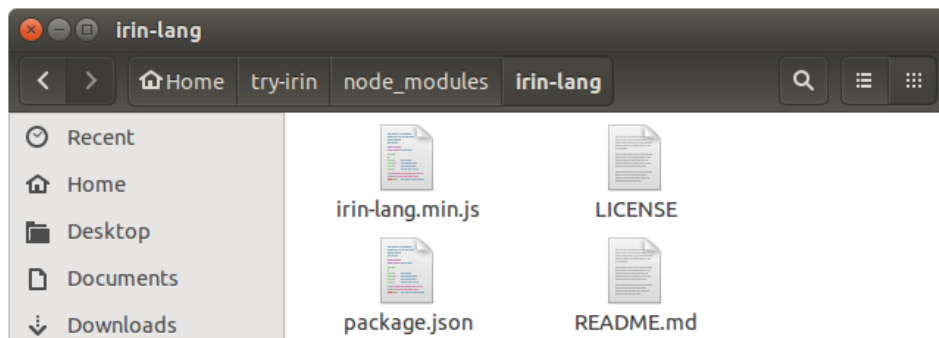
การติดตั้งผ่าน npm

หากท่านใช้งาน npm ท่านสามารถติดตั้งตัวแปลภาษาไอรินได้โดยการใช้งานคำสั่ง
npm install irin-lang



```
pakkapon@phongthawee: ~/try-irin
pakkapon@phongthawee:~/try-irin$ npm install irin-lang
try-irin@1.0.0 /home/pakkapon/try-irin
└─┬ irin-lang@0.0.4 extraneous
pakkapon@phongthawee:~/try-irin$
```

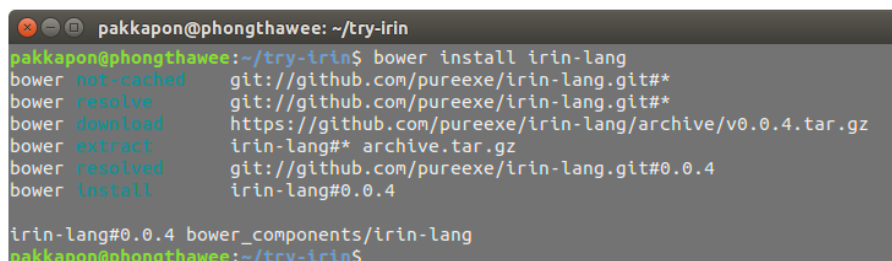
เมื่อการติดตั้งเสร็จสมบูรณ์จะปรากฏข้อความดังภาพ



จะปรากฏไฟล์ขึ้นมาที่ โฟลเดอร์/node_modules/irin-lang

การติดตั้งผ่าน bower

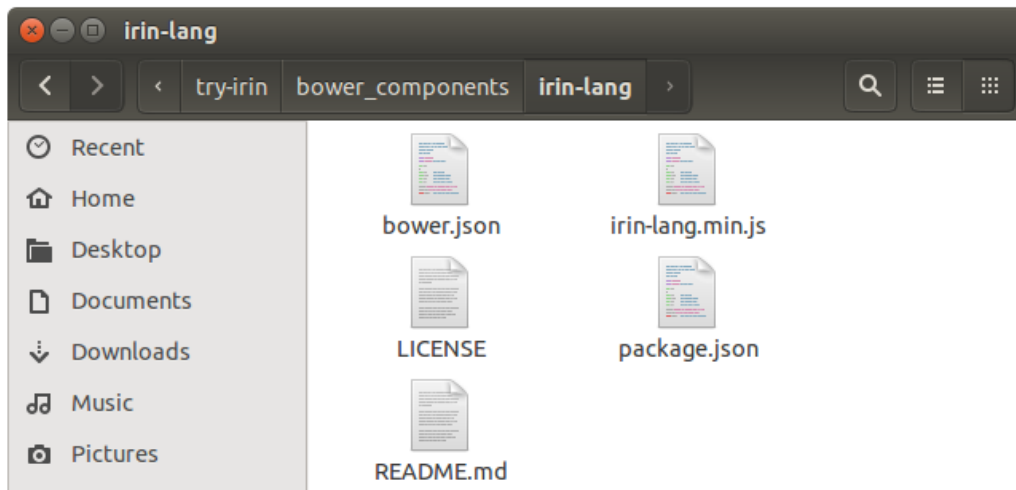
หากท่านใช้งาน bower ท่านสามารถติดตั้งตัวแปลภาษาไอรินได้โดยการใช้งานคำสั่ง
bower install irin-lang



```
pakkapon@phongthawee: ~/try-irin
pakkapon@phongthawee:~/try-irin$ bower install irin-lang
bower not-cached git://github.com/pureexe/irin-lang.git#*
bower resolve git://github.com/pureexe/irin-lang.git#*
bower download https://github.com/pureexe/irin-lang/archive/v0.0.4.tar.gz
bower extract irin-lang# archive.tar.gz
bower resolved git://github.com/pureexe/irin-lang.git#0.0.4
bower install irin-lang#0.0.4

irin-lang#0.0.4 bower_components/irin-lang
pakkapon@phongthawee:~/try-irin$
```

เมื่อการติดตั้งเสร็จสมบูรณ์จะปรากฏข้อความดังภาพ



จะปรากฏไฟล์ขึ้นที่ โฟลเดอร์/bower_components/irin-lang

การติดตั้งโดยการดาวน์โหลดไฟล์

ท่านสามารถดาวน์โหลดไฟล์ได้ที่ <https://github.com/pureexe/irin-lang/releases>

Latest release

v0.0.4
40fa5d9

Release: v0.0.4
pureexe released this 8 minutes ago

- fix: -> on difference directory
- fix: expression - and / is wrong because stack popping error.
- fix: regular expression escape
- fix: declare variable which isn't following top-down law
- fix: ### multiline comment working incorrectly ###
- add: support data<-(other_variable)+5 in header
- add: error Unexpected declaration in header.
- add: error Variable name must not start with number.
- add: error unexpected indentation
- remove: error Header must have no indent. (replace by unexpected indentation)

Downloads

irin-lang.min.js	11.4 KB
Source code (zip)	
Source code (tar.gz)	

โดยให้ท่านดาวน์โหลดไฟล์ชื่อ irin-lang.min.js เพื่อนำไปใช้งาน

การติดตั้ง

ใช้งานผ่าน Node.JS หรือ CommonJS

ในการใช้งานผ่าน Node.JS หรือ CommonJS สามารถนำเข้าตัวแปลภาษาได้ผ่านทางคำสั่ง require โดยสามารถเขียนคำสั่งได้ดังนี้

```
var Irin = require("irin-lang")
```

ใช้งานผ่านเบราว์เซอร์

ในการใช้งานผ่านเบราว์เซอร์ สามารถนำเข้าตัวแปลภาษาได้ผ่านทางแท็ก script โดยสามารถเขียนคำสั่งได้ดังนี้

```
<script src="path/to/irin-lang.min.js"></script>
```

การใช้งานในโปรแกรม

การแปลภาษา

ก่อนที่จะเริ่มโต้ตอบกับระบบสนทนาได้ตอบอัตโนมัติที่เขียนขึ้นจากภาษาไอรินได้ จำเป็นต้องทำการแปลภาษาก่อน ซึ่งสามารถทำการแปลภาษาได้โดยการเขียนโค้ดภาษาจาวาสคริปดังนี้

```
var bot = new Irin("ที่อยู่ไฟล์.irin",function(err){  
    if(err){  
        //เกิดข้อผิดพลาดขึ้นขณะแปลภาษา  
    }else{  
        //การแปลภาษาเสร็จสมบูรณ์  
    }  
});
```

การโต้ตอบ

การโต้ตอบกับโค้ดภาษาไอรินที่ถูกแปลภาษาเรียบร้อยแล้วนั้นทำได้โดยการเขียนโค้ดภาษาจาวาสคริปดังนี้

```
bot.reply("คำถาม")
```

การใช้งานภาษาไอริน

การเยื้องเพื่อการรู้จำภาษาธรรมชาติอย่างชาญฉลาด (Indent to Recognize for Intelligent Natural language : IRIN) หรือเรียกอีกชื่อว่า ภาษาไอริน เป็นภาษาที่แบบมาให้ใช้การเยื้องในการแทนตำแหน่งของคำถามและคำตอบในการสร้างระบบสนทนาโต้ตอบอัตโนมัติซึ่งจะทำให้โค้ดอ่านง่ายและสั้นลงอย่างมาก ซึ่งมีไวยากรณ์ในการเขียนดังนี้

การเยื้อง

ในภาษาไอรินใช้การเยื้องในการแทนตำแหน่งของคำถามและคำตอบโดยบรรทัดที่ไม่มีการเยื้องเลยจะทำหน้าที่เป็นคำถาม และบรรทัดที่เยื้องเข้าไปจะเป็นคำตอบ เมื่อเยื้องเข้าไปอีกจะเป็นคำถาม สลับกันไปเรื่อยๆ ตัวอย่างเช่น

```
1 | สวัสดี
2 | มีอะไรให้ช่วยหรือคะ
```

เมื่อถามระบบว่า สวัสดี ระบบจะตอบว่า มีอะไรให้ช่วยหรือคะ ซึ่งจะเห็นได้ว่า สวัสดี ทำหน้าที่เป็นคำถาม และ มีอะไรให้ช่วยหรือคะ เป็นคำตอบ

อีกทั้งภาษาไอรินยังใช้การเยื้องในการแยกคำถามแต่ละข้อออกจากกัน ตัวอย่างเช่น

```
1 | สวัสดี
2 | มีอะไรให้ช่วยหรือคะ
3 | ลาก่อน
4 | ลาก่อนค่ะ
```

จะเห็นว่าโค้ดดังกล่าวมี 2 คำถามคือ สวัสดี กับลาก่อน

แต่หากไม่มีการเยื้องในบรรทัดที่ติดกันจะถือว่าเป็นคำถามที่มีคำตอบร่วมกัน ตัวอย่างเช่น

```
1 | สวัสดี
2 | ดีจ้า
3 | หวัดดี
4 | มีอะไรให้ช่วยหรือคะ
```

จะสังเกตว่าบรรทัดที่ 1,2 และ 3 ไม่มีการเยื้อง จึงถือว่าเป็นคำถามร่วมกัน ดังนั้นเมื่อถามระบบว่า สวัสดี, ดีจ้า, หรือหวัดดี ระบบจะตอบกลับมาเหมือนกันคือ มีอะไรให้ช่วยหรือคะ

และหากไม่มีการเยื้องในบรรทัดที่ติดกันของคำตอบจะถือว่าเป็นคำตอบร่วมกันของคำถาม โดยคำตอบร่วมกัน จะใช้การสุ่มหนึ่งในคำตอบออกมาตอบ ตัวอย่างเช่น

```
1 | สวัสดี
2 | สวัสดีค่ะ
3 | ยินดีต้อนรับค่ะ
4 | มีอะไรให้ช่วยหรือคะ
```

หากเราถามระบบว่าสวัสดี ระบบอาจตอบว่า สวัสดีค่ะ, ยินดีต้อนรับค่ะ หรือมีอะไรให้ช่วยหรือคะ

ทำนองเดียวกันเราสามารถใช้คำถามร่วมและคำตอบร่วมพร้อมกันได้ ตัวอย่างเช่น

1	สวัสดี
2	ดีจ้า
3	หวัดดี
4	สวัสดีค่ะ
5	ยินดีต้อนรับค่ะ
6	มีอะไรให้ช่วยหรือคะ

เมื่อถามระบบว่าสวัสดี,ดีจ้า หรือหวัดดี ระบบอาจตอบว่า สวัสดีค่ะ, ยินดีต้อนรับค่ะ หรือมีอะไรให้ช่วยหรือคะ

อีกทั้งเรายังสามารถใช้การเยื้องเพื่อเป็นการบอกการสนทนาต่อเนื่องได้ โดยการเพิ่มการเยื้องเข้าไปต่อจากคำตอบของคำถามก่อนหน้านี้ ตัวอย่างเช่น

1	สวัสดี
2	สวัสดีค่ะ
3	นอนละ
4	ฝันดีค่ะ
5	หิวข้าว
6	ไปกินสิคะ
7	ไม่มีเงิน
8	เดี๋ยวให้ยืมคะ
9	ไม่คืนได้ไหม
10	ไม่ได้ค่ะ
11	ไม่หิวแล้ว
12	แล้วจะบอกว่าหิวข้าวทำไมคะ

สังเกตว่าบริเวณคำว่าหิวข้าวมีการเยื้องถัดเข้าไปต่อจากคำตอบของคำถามแรก นั้นหมายถึงการสนทนาต่อเนื่อง เมื่อถามระบบว่า สวัสดี ตัวแปลภาษาจะตอบว่า สวัสดีค่ะ แต่เมื่อเราถามระบบต่อด้วยคำว่าหิวข้าว ระบบจะตอบกลับมว่า ไปกินสิคะ ซึ่งช่วงนี้จะเป็นการสนทนาต่อเนื่องซึ่งตอนนี้มี 2 ทางเลือกคือ ไม่มีเงิน กับไม่หิวแล้ว ซึ่งหากถามระบบต่อว่าไม่มีเงิน ระบบจะตอบว่า เดี๋ยวให้ยืมคะ แต่หากเรากรอกว่านอนละ ระบบจะออกจากการสนทนาต่อเนื่องแล้วตอบว่า ฝันดีค่ะ

ความคิดเห็น

ความคิดเห็นหรือคอมเมนต์ (Comment) ใช้เพื่อบอกความคิดเห็นของเราลงไปในโค้ด โดยความคิดเห็นนั้นมีไว้เพื่อให้มนุษย์อ่านเท่านั้น เพื่อให้ผู้เขียนโค้ดสามารถจำได้ว่าโค้ดที่ตัวเองเขียนมีไว้ใช้งานอย่างไร โดยความคิดเห็นนั้นจะไม่ถูกนำไปประมวลผลโดยตัวแปลภาษา

ความคิดเห็นบรรทัดเดียว

ใช้เครื่องหมาย # เพื่อระบุว่าเป็นความคิดเห็นบรรทัดเดียว โดยตัวแปลภาษาจะไม่นำข้อความหลังเครื่องหมายไปประมวลผล ตัวอย่างเช่น

```
1  #นี่คือความคิดเห็น
2  สวัสดี
3  มีอะไรให้ช่วยหรือคะ
```

ความคิดเห็นหลายบรรทัด

ใช้เครื่องหมาย ### เพื่อระบุว่าเป็นความคิดเห็นหลายบรรทัด โดยต้องปิดท้ายความเห็นด้วย ### เสมอ โดยตัวแปลภาษาจะไม่นำข้อความที่อยู่ในช่วงของเครื่องหมายดังกล่าวไปประมวลผล ตัวอย่างเช่น

```
1  ###
2  สิ่งที่ปรากฏส่วนนี้
3  จะไม่ทำงาน
4  ###
5  สวัสดี
6  มีอะไรให้ช่วยหรือคะ
```

นิพจน์

นิพจน์ (Expression) คือ รูปแบบสำหรับเปรียบเทียบข้อความว่าเข้าเงื่อนไขหรือไม่ หากนิพจน์ตรงกับรูปแบบของศัพท์ที่กรอกเข้ามาจะทำการเรียกคำตอบของคำถามนั้นถัดไป โดยภาษาไอรินมีนิพจน์ต่างๆดังต่อไปนี้

ดอกจันท์

ดอกจันท์ * คือ ตรงกับทุกตัวอักษร ไม่ว่าตัวอักษรใดก็ตามจะเป็นจริงเสมอ ตัวอย่างเช่น

```
1  สวัสดี
2  สวัสดีค่ะ
3  *
4  ขอภัยค่ะ ดิฉันไม่เข้าใจ
```

เมื่อเราถามคำถามใดกับระบบก็ตามที่ไม่ใช่ สวัสดี ระบบจะตอบกลับมว่า ขอภัยค่ะ ดิฉันไม่เข้าใจ เนื่องจากนิพจน์ดอกจันท์แทนตัวอักษรใดก็ได้

นอกจากนี้เรายังใช้นิพจน์ดอกจันท์แทนบางส่วนของประโยคได้เช่น

```
1  ผมว่าเธอ*มาก
2  ทำไมคุณถึงคิดอย่างนั้นละ
```

เมื่อถามระบบว่า ผมว่าเธอน่ารักมาก กับถามระบบว่า ผมว่าเธอสวยมาก ระบบจะตอบกลับมาว่า ทำไมคุณถึงคิดอย่างนั้นละ

แต่ว่าดอกจันทน์นั้นไม่ถือว่าตรงกับคำว่าว่างเปล่า ต้องมีอย่างน้อย 1 ตัวอักษรขึ้นไป ตัวอย่างเช่นโค้ดด้านบน หากถามระบบว่า ผมว่าเธอมาก จะสังเกตได้ว่า บริเวณที่เป็นดอกจันทน์จะไม่ตรงกับตัวอักษรใดๆเลย ดังนั้นระบบจะไม่ตอบว่าทำไมคุณถึงคิดอย่างนั้นละ จึงขอให้ระวังในส่วนนี้อาไว้ด้วย

วงเล็บ

() คือ เลือก คำใดคำหนึ่งจากในวงเล็บ แบ่งคำออกจากกันด้วยเครื่องหมาย | ตัวอย่างเช่น

1	หิว(ใหม่ หรือไม่)
2	หิวแล้วค่ะ

เมื่อถามระบบว่า หิวใหม่ หรือถามว่า หิวหรือไม่ ระบบจะตอบกลับมาว่า หิวแล้วแล้วค่ะ เนื่องจากคำว่าใหม่และคำว่าหรือไม่เป็นคำใดคำหนึ่งจากในวงเล็บ แต่หากถามว่าหิวมะ ระบบจะไม่ตอบว่าหิวแล้วค่ะ เนื่องจากคำว่ามะ ไม่ได้ตรงกับคำใดคำหนึ่งในวงเล็บ

วงเล็บก้ามปู

[] คือ อาจ มีคำใดคำหนึ่งจากในวงเล็บ แบ่งคำออกจากกันด้วยเครื่องหมาย | ตัวอย่างเช่น

1	สวัสดี[ครับ ค่ะ]
2	สวัสดีค่ะ

เมื่อถามระบบว่า สวัสดี,สวัสดีครับ หรือสวัสดีค่ะ ระบบจะตอบกลับมาว่า สวัสดีค่ะ แต่หากถามระบบว่าสวัสดีนะ ระบบจะไม่ตอบกลับมาว่าสวัสดีค่ะ เนื่องจากไม่ได้บอกกับระบบว่าจากมีคำว่านะ อยู่ด้านหลัง

และเรายังสามารถใช้เครื่องหมายวงเล็บก้ามปูร่วมกับดอกจันทน์ได้ ตัวอย่างเช่น

1	สวัสดี[*]
2	สวัสดีค่ะ

นิพจน์นี้มีความหมายว่าอาจมีตัวอักษรใดก็ได้ ดังนั้นจะตรงกับข้อความที่มี หรือไม่มีตัวอักษรต่อท้ายก็ได้ ดังนั้นจึงสามารถตรงกับข้อความว่า สวัสดีจ้า สวัสดีค่ะ หรือสวัสดีก็ได้

ส่วนหัว

ส่วนหัวมีไว้สำหรับการประกาศตัวแปรโดยส่วนหัวจะเริ่มต้นด้วยขีดกลาง - 3 ขีด และปิดด้วยขีดกลางอีก 3 ขีด โดยการกำหนดตัวแปรนั้นจะใช้เครื่องหมาย <- โดยการตั้งชื่อตัวแปรนั้น สามารถตั้งโดยใช้ตัวหรือตัวอักษรก็ได้แต่ห้ามขึ้นต้นด้วยตัวเลข ตัวอย่างเช่น

```

1  ---
2  name <- ไอริน
3  gender <- หญิง
4  age <- 18
5  ---

```

โดยจากตัวอย่างโค้ดคือการประกาศตัวแปร 3 ตัว ตัวแปรแรกชื่อ name เก็บค่า ไอริน ตัวแปรที่ 2 ชื่อ gender เก็บค่าหญิง ตัวแปรที่ 3 เก็บค่า 18

การเข้าถึงตัวแปร

การเข้าถึงตัวแปรสามารถทำได้โดยใช้เครื่องหมายปีกกา { } โดยแบ่งตัวแปรเป็น 2 ประเภทได้แก่ ตัวแปรทั่วไปและตัวแปรตามเหตุการณ์

ตัวแปรทั่วไป

คือตัวแปรที่ประกาศไว้บนส่วนหัวของไฟล์ สามารถเรียกใช้งานได้โดยเขียนว่า {ชื่อตัวแปร} แล้วตัวแปลภาษาจะทำการแทนค่าของตัวแปรไปยังบริเวณที่ {ชื่อตัวแปร} ปรากฏอยู่ ตัวอย่างเช่น

```

1  ---
2  name <- ไอริน
3  ---
4  ชื่ออะไร
5  ดิฉันชื่อ{name}ค่ะ

```

เมื่อถามระบบว่า ชื่ออะไร ระบบจะตอบว่า ดิฉันชื่อไอรินค่ะ เนื่องจากตัวแปลภาษาได้แทนค่าของตัวแปร name ลงไปยังบริเวณที่เขียนว่า {name}

นอกจากนี้เรายังสามารถใช้ตัวแปรทั่วไปกับคำถามได้ด้วย ตัวอย่างเช่น

```

1  ---
2  name <- ไอริน
3  ---
4  {name}
5  เรียกดิฉันทำไมคะ

```

เมื่อเราถามระบบว่า ไอริน ระบบจะตอบกลับว่า เรียกดิฉันทำไมคะ เนื่องจากค่าของตัวแปร name ซึ่งมีค่าเป็นไอริน จะถูกแทนที่ลงในคำถาม ดังนั้นคำถามนั้นจึงเป็นคำว่า ไอริน ทำให้เมื่อถามว่าไอรินแล้วระบบจึงตอบว่า เรียกดิฉันทำไมคะ

ตัวแปรตามเหตุการณ์

ตัวแปรตามเหตุการณ์จะเกิดขึ้นเมื่อมีการใช้นิพจน์ () และ * เนื่องจากนิพจน์ 2 ตัวนี้จะต้องมีค่าใดค่าหนึ่งแน่นอน โดยการเข้าถึงจะใช้ {ตำแหน่งของนิพจน์} โดยนับเป็นเลขเริ่มต้นจาก 1 ตัวอย่างเช่น

```
1  ผม(ชอบ|เกลียด)*
2  ทำไมคุณถึงได้{1}พวก{2}ละ
```

เมื่อถามระบบว่า ผมชอบแมว ระบบจะตอบกลับมาว่า ทำไมคุณถึงชอบพวกแมวละเนื่องจากคำว่าชอบ ถูกนำไปแทนค่าในตัวแปร {1} และคำว่า แมว ถูกนำไปแทนค่าในตัวแปร {2}

การเปลี่ยนค่าตัวแปร

เราสามารถเปลี่ยนแปลงค่าของตัวแปรแบบทั่วไป ที่มีอยู่เดิมได้การกำหนดค่าตัวแปรเข้าไปใหม่แบบพลวัตโดยการใช้เครื่องหมาย <- เพื่อกำหนดค่าตัวแปรเข้ามาใหม่ภายใต้เครื่องหมายปีกกา ตัวอย่างเช่น

```
1  ---
2  name <- ไอริช
3  isHungry <- กำลังหิวเลยล่ะ
4  ---
5  กินข้าวกัน
6  {name}{isHungry}{isHungry<-อิ่มแล้วล่ะ}
```

จากตัวอย่าง เมื่อถามระบบว่า กินข้าวกัน ระบบจะตอบกลับมาว่า ไอริชกำลังหิวเลยล่ะ เมื่อถามระบบอีกครั้งว่า กินข้าวกัน ระบบจะตอบกลับมาว่า ไอริชอิ่มแล้วล่ะ

โดยการกำหนดค่านั้นจะทำเมื่อเจอวงเล็บปีกกาที่มีเครื่องหมาย <- ดังนั้นหากเราสลับตำแหน่งของวงเล็บปีกกาในโค้ดตัวอย่างด้านบน {name}{isHungry<-อิ่มแล้วล่ะ}{isHungry} เมื่อถามว่ากินข้าวกัน ระบบจะตอบกลับมาว่า ไอริชอิ่มแล้วล่ะ เนื่องจากโค้ดดังกล่าวได้ทำการกำหนดค่าตัวแปรใหม่ก่อนนำไปแสดงผล

อีกทั้งภายใต้เครื่องหมายปีกกายังสามารถกำหนดค่าได้โดยการใช้เครื่องหมายปีกกา ยังสามารถใช้เครื่องหมาย บวก ลบ คูณ หาร ในการช่วยกำหนดค่าได้ ตัวอย่างเช่น

```
1  ---
2  sheep <- 0
3  ---
4  นับแกะ
5  {sheep<-{sheep}+1}แกะ{sheep}ตัว
```

หากถามระบบว่านับแกะ ระบบจะตอบว่า แกะ1ตัว หากถามระบบอีกครั้งว่า นับแกะ ระบบจะตอบว่าแกะ 2 ตัวและเพิ่มขึ้นเรื่อยๆตามลำดับ

สังเกตว่าคำสั่ง {sheep<-{sheep}+ 1} จำเป็นต้องใส่วงเล็บปีกกาที่ชื่อตัวแปรในฝั่งขวาของเครื่องหมาย <- เสมอ หากไม่ใส่เครื่องหมายปีกการะบบจะถือว่าคำว่า sheep เป็นอักขระ ไม่ใช่ตัวแปร

และเรายังสามารถนำค่าที่ได้มาจากตัวแปรตามเหตุการณ์กำหนดค่าใส่ในตัวแปรทั่วไปเพื่อใช้งานค่าในภายหลังได้ ดังตัวอย่าง

```
1  ---
2  user <- คุณ
3  ---
4  สวัสดี
5      สวัสดีค่ะ คุณชื่ออะไรหอรคะ
6      [ผม|ฉัน][ชื่อ]*[ครับ|ค่ะ]
7      สวัสดีค่ะคุณ{1}{user<--{1}}
8  หิวข้าว
9      ทำไม{user}ไม่ไปกินข้าวละ
```

เมื่อถามระบบว่า หิวข้าว ระบบจะตอบว่า ทำไมคุณไม่ไปกินข้าวละ เมื่อถามระบบต่อว่า สวัสดี ระบบจะตอบว่า สวัสดีค่ะ คุณชื่ออะไรหอรคะ จึงถามระบบต่อว่า ผมชื่อเพียวครับ ระบบตอบกลับมว่า สวัสดีค่ะคุณเพียว และเมื่อถามระบบต่อว่า หิวข้าว ระบบจะตอบว่า ทำไมเพียวไม่ไปกินข้าวละ จะสังเกตว่าคำถามว่าหิวข้าวทั้ง 2 ครั้งชื่อผู้ใช้จะแตกต่างกัน

เงื่อนไขในบรรทัด

เราสามารถทำให้คำตอบการสนทนาเปลี่ยนไปได้โดยการเงื่อนไขในบรรทัดได้ โดยใช้เครื่องหมาย ? ตามด้วยคำตอบเมื่อเป็นจริง แบ่งด้วย : แล้วตามด้วยเงื่อนไขเมื่อเป็นเท็จ โดยเครื่องหมายในการเปรียบเทียบที่สามารถใช้ได้ ได้แก่ และ (&&), หรือ (||), นิเสธ (!), เท่ากับ (==), มากกว่าเท่ากับ (>=), น้อยกว่าเท่ากับ (<=), มากกว่า (>), น้อยกว่า (<), บวก(+), ลบ (-), คูณ (*) และหาร (/) ตัวอย่างเช่น

```
1  เธออายุเท่าไร
2      อายุ 18 ค่ะ แล้วคุณอายุของคุณเท่าไรคะ
3      [ผม|ฉัน][อายุ]*[ครับ|ค่ะ]
4      คุณนี่ดู{1}<18?เด็ก:แก่}จังเลยคะ
```

เมื่อถามระบบว่า เธออายุเท่าไร ระบบจะตอบว่า อายุ 18 ค่ะ แล้วคุณอายุของคุณเท่าไรคะ เมื่อถามกลับไว่า อายุ 20 ครับ ระบบจะตอบกลับมว่า คุณนี่ดูแก่จังเลยคะ แต่หากถามกลับไว่า อายุ 12 ครับ ระบบจะตอบว่า คุณนี่ดูเด็กจังเลยคะ

หัวข้อเรื่อง

ในภาษาไอริน ฟังก์ชันนั้นจะใช้เครื่องหมาย -> แทนการกำหนดหัวข้อเรื่อง ทั้งการประกาศหัวข้อเรื่อง และการนำเข้าสู่หัวข้อเรื่อง โดยการประกาศและใช้หัวข้อเรื่องจะ -> ชื่อหัวข้อ ต่างกันที่การประกาศหัวข้อเรื่อง นั้นนั้นก่อนหน้าเครื่องหมาย -> จะไม่มีการเยื้องเลย หากมีการเยื้องจะเป็นการนำเข้าสู่หัวข้อเรื่องนั้น ตัวอย่างเช่น

```
1 | สวัสดี
2 |   สวัสดีค่ะ
3 |   ->จากลา
4 | ->จากลา
5 |   ไปละ
6 |   ลาก่อนค่ะ
```

เมื่อถามระบบว่า สวัสดี ระบบจะตอบว่า สวัสดีค่ะ หลังจากนั้นถามว่า ไปละ ต่อระบบจะตอบว่า ลาก่อนค่ะ เนื่องจากเยื้องของคำว่าสวัสดีค่ะ มีการเรียกใช้หัวข้อจากลา ซึ่งจะเป็นการดึงเอาคำถามและคำตอบที่อยู่ในหัวข้อจากลาไปเชื่อมอยู่กับคำถามสวัสดี ดังนั้นคำถามที่อยู่ต่อจากคำว่าสวัสดีค่ะก็คือคำว่าไปละ หรือก็คือผลลัพธ์ที่ได้จะเหมือนกับการเขียนโค้ดดังนี้

```
1 | สวัสดี
2 |   สวัสดีค่ะ
3 |   ไปละ
4 |   ลาก่อนค่ะ
```

และยังสามารถเรียกหัวข้อจากไฟล์อื่นได้โดยการนำหัวข้อเรื่องใส่ไว้ในไฟล์อื่นแล้วตั้งชื่อไฟล์ด้วยนามสกุล .irin แล้วทำการเรียกใช้ด้วยคำสั่งหัวข้อเรื่อง

ตัวอย่างเช่น ไฟล์แรกเขียนว่า

```
1 | สวัสดี
2 |   สวัสดีค่ะ
3 |   ->goodbye.irin
```

ไฟล์ goodbye.irin เขียนว่าให้ทำการเขียนประโยคการโต้ตอบได้เลยโดยไม่ต้องใส่ชื่อหัวข้อเรื่อง

```
1 | ไปละ
2 |   ลาก่อนค่ะ
```

จะเห็นว่าเมื่อรันแล้วผลลัพธ์จะสามารถใช้งานได้เหมือนกับการเขียนหัวข้อเรื่องไว้ในไฟล์เดียวกัน

แต่หากการเขียนเครื่องหมาย -> โดยไม่มีการเยื้องและตามด้วย .irin นั้นถือเป็นการนำเข้าไฟล์มาใช้งานไม่ใช่การประกาศหัวข้อเรื่อง ตัวอย่างเช่น

```

1 | สวัสดี
2 |   สวัสดีค่ะ
3 | ->goodbye.irin

```

ผลลัพธ์ที่ได้จะทำงานเหมือนกับโค้ดดังต่อไปนี้

```

1 |   สวัสดี
2 |   สวัสดีค่ะ
3 |   ไปละ
4 |   ลาก่อนค่ะ

```

บนลงล่าง

สำหรับภาษาไอรินจะมีการเรียงลำดับความสัมพันธ์โดยการทำงานโค้ดจากบนลงล่าง หากโค้ดที่มีลำดับการเยื้องเดียวกันจะทำการเลือกโค้ดที่อยู่ด้านบนเสมอ ตัวอย่างเช่น

```

1 |   สวัสดี
2 |   สวัสดีค่ะ
3 |   สวัสดี
4 |   สวัสดีจ้า

```

เมื่อถามระบบว่าสวัสดี ระบบจะตอบว่าสวัสดีค่ะ เสมอเนื่องจากสวัสดีในบรรทัดที่หนึ่งเป็นบรรทัดแรก ที่ตรงกับเงื่อนไข ดังนั้นจึงตอบว่าสวัสดีค่ะเสมอ

นอกจากนี้การทำงานจากบนลงล่างยังมีผลต่อการกำหนดค่าตัวแปรอีกด้วยตัวอย่างเช่น

```

1 | ---
2 | name <- ไอริน
3 | name <- อลิส
4 | ---

```

จากโค้ดตัวอย่างมีการประกาศตัวแปรชื่อ name ซ้ำกัน 2 ครั้งดังนั้นแล้วค่าของตัวแปรจะเป็น ไอริน เนื่องจากโค้ดพบว่า

การกำหนดหัวข้อชื่อซ้ำกันจตัวแปลภาษาจะเลือกใช้งานหัวข้อที่เจอเป็นหัวข้อแรก ตัวอย่างเช่น

```

1 |   หิวข้าวไหม
2 |   หิวแล้วค่ะ
3 |   ->ไปกินข้าว
4 |
5 | ->ไปกินข้าว
6 |   ไปกินข้าวกัน
7 |   ได้เลยค่ะ
8 |
9 | ->ไปกินข้าว
10 |   ไปกินข้าวกัน
11 |   ไววันหลังนะคะ

```

เมื่อถามระบบว่า หิวข้าวไหม ระบบตอบว่าหิวแล้วค่ะ เมื่อถามระบบต่อว่า ไปกินข้าวกัน ระบบจะตอบว่าได้เลยค่ะเสมอ เนื่องจากเจอหัวข้อเรื่องที่บรรทัดที่ 5 ก่อน

ประโยชน์ของการใช้งานโค้ดแบบบนลงล่างคือไม่สามารถเขียนทับตัวแปรที่ของไฟล์อื่นได้เหมาะสมกับกรณีไฟล์ภาษาไอรินมาจากการเขียนโดยคนละคนกันตัวอย่างเช่น

```
1  ---
2  name <- ไอริน
3  ---
4
5  -> default.irin
6  -> aichan/main.irin
7  -> alice/main.irin
8  -> arisa/main.irin
```

ถ้าทั้ง 4 ไฟล์เขียนโดยคนละคนกันแล้วใช้ ในแต่ละไฟล์มีการประกาศ ตัวแปรชื่อ name เหมือนกัน ตัวแปรทั้งหมดนั้นจะถูกเปลี่ยนค่าเป็น ไอรินทันทีเพื่อการอำนวยความสะดวกในการเขียนโค้ดของท่าน

นอกจากนี้แล้วท่านยังสามารถอ่านเอกสารวิธีการใช้งานภาษาไอรินและตัวแปลภาษาไอริน เพิ่มเติมได้ที่ <https://ภาษา.ไอริน.ไทย> และสามารถลองใช้งานภาษาไอรินออนไลน์ได้ที่ <https://ภาษา.ไอริน.ไทย/ลองใช้>